Home Assignment (Model checking)

Title: **Cache Coherence**

Advisor: **Ákos Hajdu**

Problem Description

The purpose of the cache is to keep a copy of parts of the memory frequently used by the CPU to boost access speed. In multi-processor systems, the main memory and the caches are connected to a memory bus, and a CPU can only access the contents of the main memory through its own cache. The task is to ensure coherence, i.e. that all CPUs see identical memory contents.

The smallest unit handled by caches is called a *cache line* (e.g. 64 bytes). Due to the small size of caches, the copies of most cache lines are not in the cache (*invalid* state). The exact copy of a memory area found in a CPU's cache can often occur in other CPU's caches at the same time (*shared* state). If the cache line is in *exclusive* state, that line should be *invalid* in the cache of every other CPU. Finally, a cache line can be in *dirty* state (which is also *exclusive*), meaning that it has been modified locally in the cache, and needs to be written back into the main memory.

If a CPU wants to read from a shared cache line, it can be served locally (without turning to the memory bus). Writing a dirty or exclusive line can also be served locally (but exclusive states become dirty upon writing). If a CPU wants to write into a shared cache line, the cache also writes the value to the main memory and it shifts to exclusive state. All the other caches detect the change and invalidate their incidental local copies of this line.

If the currently needed line is missing from the cache (*invalid*) and a CPU wants to read, first it signals the reading request on the memory bus. This makes every *exclusive* or *dirty* copy *shared*, while writing the locally modified value into the main memory as well. After that, the line is copied into the initiating CPU's cache in shared state. In order to write an *invalid* cache line it has to be read first (because writing usually affects only a part of the line), then the writing process is similar to *shared* writing described above. Sometimes a cache line is forced out of the cache due to lack of space, when another *invalid* line is read or written (thus added to the cache). Then the eliminated line becomes *invalid*, and if it was *dirty,* the main memory is refreshed beforehand.

Requirements to check

Model the local states of *one cache line with 4 CPUs*. The domain of the cache line can be restricted to a small domain, e.g., a bounded integer ranging from 0 to 2. When a CPU writes its cache, use a random value (Select construct). Prove the satisfiability of the requirements below using temporal logic expressions and model checking (in case of unsatisfiability explain the reasons in detail with a counterexample)! Show and explain a short example/counterexample where possible!

1. The system has no deadlocks.
2. The cache line can be shared between all the caches at the same time.
3. If the cache line is exclusive or dirty in the *first* CPU, the other CPUs cannot have a copy.
4. All exclusive and shared copies are identical to the value in the main memory.